MetaRule

Random

Sean Barnum, Cigital, Inc. [vita[1]]

Copyright © 2007 Cigital, Inc.

2007-03-29

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 5971 bytes

Attack Category	• Encryption Assault
Vulnerability Category	• Random number problems
Software Context	• Cryptography
Location	
Description	The random function is a Linear Congruential Generator (LCG) that is used to create pseudorandom integers. That by itself is not a security breach. However, how the numbers are used can be a problem. The algorithm that generates the numbers is well known, the range of numbers generated is very small (in a cryptographic context), and the generated numbers can be guessed with reasonable ease. Hence, if the pseudorandom numbers are used as the basis for encryption computations, then it becomes a security problem. There is simply not enough randomness or entropy in pseudorandom numbers generated by LCGs for them to be used in high security encryption.
APIs	**Function Name** / **Comments** drand48 erand48 initstate jrand48 lcong48 lrand48 mrand48 nrand48 random seed48 setstate

1. http://buildsecurityin.us-cert.gov/bsi-rules/35-BSI.html (Barnum, Sean)

srand	
srand	
x	
srandom	
rand	

Method of Attack

Exception Criteria

Solutions

Solution Applicability	Solution Description	Solution Efficacy
Applicable to all occurrences	Most if not all cryptographic toolkits include cryptographically secure pseudorandom number generators (PRNG). One of these secure PRNGs should be used instead of LCGs. However, the secure PRNGs are only effective if they are "seeded" with a truly random value that is impossible to guess (except with a negligible probability). The random seed should be obtained using any utility provided by the operating system. For example, many UNIX variants provide /dev/random and /dev/urandom, whereas Microsoft	Effective

	Windows provides CryptGenRandom().
Signature Details	double drand48(void) double erand48(unsigned short int xsubi[3]) char *initstate(unsigned int seed, char *state, size_t size) long int lrand48(void) void lcong48(unsigned short int param[7]) long int lrand48(void) long int mrand48(void) long int nrand48(unsigned short int xsubi[3]) long int random(void) unsigned short *seed48(unsigned short seed16v[3]) char *setstate(char *arg_sate) void srand(unsigned int seed) void srand48(long int seedval) void srandom(unsigned int seed) int rand (void)
Examples of Incorrect Code	`srand(time(NULL));` `key = rand();`
Examples of Corrected Code	`// Depends on package used`
Source References	• Viega, John & McGraw, Gary. *Building Secure Software: How to Avoid Security Problems the Right Way*. Boston, MA: Addison-Wesley Professional, 2001, ISBN: 020172152X, ch. 10. • Rough Auditing Tool for Security (RATS)[2]
Recommended Resource	Michael Howard's Web Log[3]
Discriminant Set	**Operating Systems**: • Windows • UNIX **Languages**: • C • C++

Cigital, Inc. Copyright

Copyright © Cigital, Inc. 2005-2007. Cigital retains copyrights to this material.

Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and “No Warranty” statements are included with all reproductions and derivative works.

For information regarding external or commercial use of copyrighted materials owned by Cigital, including information about “Fair Use,” contact Cigital at copyright@cigital.com[1].

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

1. mailto:copyright@cigital.com